Expecting the Unexpected: Fast and Reliable Detection of Missing RFID Tags in the Wild

Muhammad Shahzad and Alex X. Liu
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan, USA
{shahzadm, alexliu}@cse.msu.edu

***Abstract*—RFID systems have been deployed to detect missing products by affixing them with cheap passive RFID tags and monitoring them with RFID readers. Existing missing tag detection protocols require the tag population to contain only those tags whose IDs are already known to the reader. However, in reality, tag populations often contain tags with unknown IDs, called unexpected tags, and cause unexpected *false positives i.e.*, due to them, missing tags are detected as present. We take the first step towards addressing the problem of detecting the missing tags from a population that contains unexpected tags. Our protocol, RUN, mitigates the adverse effects of unexpected false positives by executing multiple frames with different seeds. It minimizes the missing tag detection time by first estimating the number of unexpected tags and then using it along with the false positive probability to obtain optimal frame sizes and number of times Aloha frames should be executed. RUN works with multiple readers with overlapping regions. It is easy to deploy because it is implemented on readers as a software module and does not require modifications to tags or to the communication protocol between tags and readers. We implemented RUN along with four major missing tag detection protocols and the fastest tag ID collection protocol and compared them side-by-side. Our experimental results show that RUN always achieves the required reliability whereas the best existing protocol achieves a maximum reliability of only 67%.**

I. Introduction

A. Background & Motivation

Shoplifting, employee theft, and vendor fraud have become major causes of lost capital for retailers [14]. In 2011 alone, the retailers lost an estimated 34.5 billion dollars due to these causes [1]. With the benefits of not requiring a line-of-sight and low cost of tags (*e.g.*, 5 cents per tag [11]), radio frequency identification (RFID) systems have been deployed for monitoring products by affixing them with cheap passive RFID tags and using RFID readers, which are given the IDs of the tags that are being monitored, to detect any missing tags. A tag is a microchip with an antenna in a compact package that has limited computing power and communication range. There are two types of tags: (1) passive tags, which power up by harvesting the radio frequency energy from readers (as they do not have their own power sources) and have communication range often less than 20 feet; (2) active tags, which have their own power sources and have relatively longer communication range. A reader has a dedicated power source with significant computing power. It transmits queries to a set of tags and the tags respond over a shared wireless medium. In this paper, we deal with both passive and active RFID tags.

B. Summary & Limitations of Prior Art

There are two types of missing tag detection protocols: *probabilistic* [9], [15] and *deterministic* [7], [8], [18]. The probabilistic protocols are faster but only report the event that some tags are missing, without pinpointing exactly which ones. The deterministic protocols return IDs of all the missing tags but are comparatively slower. Both approaches have their merits. In fact, they are complementary to each other, and should be used together. For example, a probabilistic protocol should be used to detect a missing tag event and once detected, a deterministic protocol should be invoked to identify which tags are missing. Several probabilistic protocols such as TRP [15] and EMTD [9] and deterministic protocols such as IIP [7], MTI [18], and SFMTI [8] have been proposed.

There are two key limitations of existing protocols. The first limitation is that all existing protocols assume a perfect environment with no unexpected tags, which is not a realistic assumption. In reality, tag populations often contain unexpected tags whose IDs are unknown. Here we give three examples. For the first example, in airports where an airline company uses RFID readers to monitor baggage of its passengers, the tags of other airline's baggage, which are in the vicinity of this airline's readers, also respond to the queries of this airline's readers. For the second example, in a large warehouse rented to multiple tenants, one tenant's RFID readers receive responses from tags of other tenants. For the third example, in a retail store that uses RFID readers to monitor only expensive merchandize, the readers receive responses from tags of inexpensive merchandize as well. Similar scenarios exist in other settings such as hospitals and malls. Existing protocols can not handle the presence of unexpected tags because they fill up unexpected slots in Aloha frames resulting in unexpected false positives.

The second major limitation of existing protocols is that except TRP, none of them is compliant with the EPCGlobal Class 1 Generation 2 (C1G2) RFID standard [4]. These protocols require the manufacturers to put random bit sequences in tags for calculating specialized hash functions. They also require the tags to be able to receive and interpret "pre-vector" and/or "post-vector" frames to select slots in frames. Such functionalities are not provisioned in the C1G2 standard because tags, especially the passive ones, do not have enough computational power. It is important for an RFID protocol to be compliant with the C1G2 standard because the cheap commercially available off-the-shelf (COTS) tags follow the C1G2 standard. A protocol that is not compliant with the

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

C1G2 standard will require home brewed tags, which will not only cost more but will also work only in limited settings. For example, if an airline uses a protocol and tags that are non-compliant with the C1G2 standard, it may be able to track its baggage at its home airport but not at the airports in rest of the world, which support only the C1G2 compliant tags.

C. Problem Statement & Proposed Approach

Now we formally define the missing tag detection problem. Let $\mathbb{E}$ represent the set of IDs of the expected tags, *i.e.*, the tags that are expected to be present in a population and need to be monitored. Let an unknown number of tags, m, out of these $|\mathbb{E}|$ tags be missing, where $0 \leq m \leq |\mathbb{E}|$. Let $\mathbb{E}_p$ be the set of IDs of the remaining $|\mathbb{E}| - m$ tags that are actually present in the population. Let $\mathbb{U}$ be the set of IDs of all the unexpected tags in the population that do not need to be monitored. We neither know exactly which IDs belong to sets $\mathbb{E}_p$ and $\mathbb{U}$ nor do we know their sizes, but we do know that $\mathbb{E}_p \subseteq \mathbb{E}$. Let T be a threshold on the number of missing tags. Our objective is to design a missing tag detection protocol using which *a set of readers should quickly detect a missing tag event with a probability* $\geq \alpha$ *whenever the number of missing tags* m *is greater than or equal to the threshold* T, where α is called the required reliability and lies in the range $0 \leq \alpha < 1$. Additionally, a missing tag detection protocol should work in single as well as multiple-reader environments, and should be compliant with the C1G2 standard.

For the problem of detecting missing tags in the presence of unexpected tags, there are three seemingly obvious solutions based on previous work. The first solution is to repeatedly execute a tag collection protocol to collect IDs of all tags and compare them with the IDs in set $\mathbb{E}$ to detect if any tags are missing. This solution works; however, it is too slow. For example, our experimental results show that even the fastest existing tag collection protocol TH [13] is 14.3 times slower than our scheme. The second solution is to first execute a tag collection protocol to get the IDs of unexpected tags and then repeatedly execute an existing missing tag detection protocol. This solution has two limitations. First, it is slow because the missing tag detection protocol will have to monitor the unexpected tags in addition to the expected tags. Second, the missing tag detection protocol will report a missing tag event even when some unexpected tags go missing, which is not the requirement. Furthermore, both these solutions can not be used in settings where readers are not allowed to read the IDs of tags in set $\mathbb{U}$ due to privacy reasons. An example of such a setting is the aforementioned multi-tenant warehouse, where one tenant may not permit readers of other tenants to read the IDs of its tags. The third solution is to repeatedly execute a tag estimation protocol and look for a net change in the population size. The limitation of this solution is that if some expected tags go missing but an equal or greater number of unexpected tags join the population, the estimation protocol can not detect the missing tag event. Furthermore, missing tag detection protocols are much faster compared to estimation protocols due to the knowledge of set $\mathbb{E}$ [7], [8], [15].

In this paper, we propose a new protocol called RFID monitoring protocol with unexpected tags (RUN), the first protocol that can achieve required reliability in detecting a missing tag event when unexpected tags might be present in the population. RUN uses the frame slotted Aloha protocol specified in the C1G2 standard as its MAC layer communication protocol. In Aloha protocol, the reader first tells the tags a frame size f and a random seed number R. Each tag within the transmission range of the reader then uses f, R, and its ID to select a slot in the frame by evaluating a hash function $h(f, R, ID)$ whose result is uniformly distributed in $[1, f]$. Each tag has a counter initialized with the slot number it chose to reply. After each slot, the reader first transmits an end of slot signal and then each tag decrements its counter by one. In any given slot, all the tags whose counters equal 1 respond with a random sequence called RN16. If no tag replies in a slot, it is called an *empty slot*. If one or more tags reply in a slot, it is called a *nonempty slot*. As per the C1G2 standard, tags do not transmit their IDs unless the reader specifically asks them to do so. In RUN, reader checks if a slot is empty or nonempty using the RN16 sequence and never asks tags to transmit their IDs. This preserves the privacy in settings where a reader is not allowed to read IDs of tags in set $\mathbb{U}$.

To detect if any tags are missing, RUN executes multiple Aloha frames with different seeds. In each frame, each tag uses the seed for that frame to select its slot. As RUN already knows the IDs of all tags in set $\mathbb{E}$, it pre-computes which tags in $\mathbb{E}$ will select which slots in each frame. Thus, it knows which slots in the frames should be nonempty if all the tags in $\mathbb{E}$ are present in the population. When a reader executes a frame, RUN compares the response in each slot of that frame with the corresponding slot in the pre-computed frame. If it finds that a particular pre-computed slot was nonempty but the corresponding slot in the executed frame is empty, it stops and declares that some tags are missing. To minimize the effect of unexpected false positives and consequently the detection time, RUN estimates the size of $\mathbb{U}$ implicitly without running an extra estimation phase and uses this estimate to calculate optimal values of system parameters. RUN works in single as well as multiple readers environment.

D. Technical Challenges & Solutions

There are three key technical challenges in detecting a missing tag event. The first technical challenge is to handle the presence of unexpected tags. Due to the presence of such tags, it is possible that a particular slot that RUN expected to be nonempty due to a specific tag in $\mathbb{E}$ actually turns out to be nonempty even though that specific tag in $\mathbb{E}$ was missing. To address this challenge, RUN executes multiple frames with different seeds, which reduces the effects of such unexpected *false positives*. We calculate the false positive probability due to tags in $\mathbb{E}_p \cup \mathbb{U}$ and use it to calculate optimal values of frame sizes and the number of times the frames should be executed to mitigate the effects of false positives.

The second technical challenge is to estimate the number of unexpected tags $|\mathbb{U}|$ in the population, which is required to calculate the optimal values of system parameters. To address this challenge, RUN first pre-computes which slots in each frame will the tags in $\mathbb{E}$ not select. Second, it executes the frames and sees how many of such slots turn out to be nonempty. The number of such slots that are nonempty in the executed frames is a function of $|\mathbb{U}|$ but is independent of $|\mathbb{E}|$ because we know from the pre-computed frames that the tags in $\mathbb{E}$ never select these slots. Thus, by observing number of

slots that are empty in the pre-computed frames and nonempty in the executed frames, RUN estimates $|\mathbb{U}|$. Note that RUN does not carry out a separate estimation phase to estimate the size of $\mathbb{U}$. It obtains the estimate while executing the Aloha frames for detecting a missing tag event and thus, does not incur any extra time cost.

The third technical challenge is to achieve the required reliability in smallest possible time. To address this challenge, we use the false positive probability to derive a "reliability condition", which, if satisfied by the system parameters, guarantees that RUN will achieve the required reliability. These values of system parameters ensure with probability α that there will be at least one slot in all the frames that is nonempty in the pre-computed frames and empty in the executed frames when $m \geq T$. To minimize RUN's execution time, we express the time in terms of the system parameters and minimize it under the constraint that the system parameters satisfy the reliability condition.

E. Key Novelty & Advantages over Prior Art

The key novelty of this paper is twofold. First, we identify the problem of detecting missing tags in the practical scenario where unexpected tags are present. Second, we propose RUN for detecting missing tags in the presence of unexpected tags. RUN has two key advantages over prior art. First, it achieves the required reliability in the presence of unexpected tags, whereas none of the existing protocols achieves the required reliability. We have extensively evaluated and compared RUN with four state-of-the-art missing tag detection protocols (TRP [15], IIP [7], MTI [18], and SFMTI [8]) in a variety of scenarios for a large range of tag population sizes. Among existing protocols, SFMTI achieves the highest reliability of 67% whereas RUN achieves arbitrarily high reliability as per the requirement. Second, it is compliant with the C1G2 standard whereas existing protocols, except TRP, are not.

II. Related Work

Several probabilistic [9], [15] and deterministic [7], [8], [18] missing tag detection protocols have been proposed. The common and major drawback of all of these protocols is that none of them handle unexpected tags and assume that the readers already know the IDs of all tags that can be present in the population. Next, we review the existing probabilistic and deterministic protocols.

A. Probabilistic Protocols

The objective of the probabilistic protocols is to detect if any tags in the population are missing. Tan *et al.* proposed the first probabilistic protocol called TRP [15]. TRP pre-computes slots in a frame and compares them with the executed slots to detect missing tags. The difference with RUN, however, lies in that TRP does not consider false positives from unexpected tags. Furthermore, for large populations, TRP requires frame size that exceeds the C1G2 specified upper limit of 2^{15}, which is not possible in practical RFID systems. Among existing protocols, TRP is the only one that is compliant with the C1G2 standard as long as the frame size is below 2^{15}. Luo *et al.* proposed another probabilistic protocol called EMTD [9]. This protocol is non-compliant with the C1G2 standard because it assumes the RFID tags to be intelligent with enough computing power to implement a hash ring and calculate hashes using that ring. None of the existing probabilistic protocols have been designed to work in multiple-reader environment.

B. Deterministic Protocols

The objective of the deterministic protocols is to identify exactly which tags are missing from a population. Li *et al.* proposed a suite of protocols in [7] out of which IIP performs the best. IIP is non-compliant with the C1G2 standard due to following three reasons. First, it requires tags to interpret pre-vector frames and reply to the reader queries as described in those frames. Second, it requires frame sizes greater than 2^{15} for large populations. Last, it requires manufacturers to insert a ring of random bits in tag memory at the time of manufacturing. IIP does not handle multiple readers either. Zhang *et al.* proposed a deterministic protocol called MTI [18], which is essentially a tag collection protocol that first collects IDs of all tags and then checks which tags are missing. MTI cannot be used to achieve an arbitrary desired accuracy because the authors do not provide a frame work to calculate system parameters. Liu *et al.* proposed a deterministic protocol called SFMTI [8]. SFMTI is non-compliant with the C1G2 tags because it requires tags to interpret non-standardized vectors before and after selecting a slot in a frame.

III. System Model

A. Architecture

For detecting missing tags, RUN uses a central controller connected with a set of readers that cover the area where the tags in set $\mathbb{E}$ are located. The use of a central controller ensures that all readers use consistent values of frame sizes and seeds when executing frames, which helps in efficiently aggregating and processing information returned by the readers. The readers use the standardized frame slotted Aloha protocol to communicate with tags and never ask the tags to transmit their IDs. The use of multiple readers with overlapping coverage regions introduces following two problems: (1) scheduling the readers such that no two readers with overlapping regions transmit at the same time, and (2) mitigating the effect of some tags responding to multiple readers due to overlap in the coverage region of those readers. For the first problem, the controller uses one of the several existing reader scheduling protocols [16] to avoid reader-reader collisions. For the second problem, we propose solution in Section IV.

B. C1G2 Compliance

RUN does not require any modifications to tags or readers. It only requires the readers to receive the frame size, persistence probability, and seed number from the controller and communicate the responses in the frames back to the controller. Persistence probability p is the probability with which a tag decides whether it will participate in a frame or not before selecting a slot in that frame. Later in the paper, we will show how we use p to handle frame sizes that exceed the C1G2 specified upper limit of 2^{15}. Such large frame sizes are required when the size of tag population is large, required reliability α is high, or the threshold T is small. As the C1G2 standard does not specify the use of p, COTS tags do not support

it. To avoid making any modifications to tags, in RUN, the reader implements p by announcing a frame size of f/p but terminating the frame after the first f slots, which can be done as per the C1G2 standard.

C. Communication Channel

We assume that the communication channel between readers and tags is reliable *i.e.*, tags correctly receives queries from the readers and the readers correctly detect transmission of RN16 sequence in a slot if one or more tags in the population transmit in that slot. If the channel is unreliable, the solution proposed in [13] can be easily adapted for use with RUN.

D. Formal Development Assumption

To make the formal development tractable, we assume that instead of picking a single slot to transmit at the start of i^{th} frame of size f, a tag independently decides to transmit in each slot of the frame with probability $1/f$ regardless of its decision about previous or forthcoming slots. Vogt first used this assumption for the analysis of Aloha protocol for RFID and justified its use by recognizing that this problem belongs to a class of problems called *occupancy problem*, which deals with the allocation of balls to urns [17]. Ever since, the use of this assumption has become a norm in the formal analysis of all Aloha based RFID protocols [12], [17], [19].

The implication of this assumption is that a tag can end up choosing more than one slots in the same frame or even not choosing any at all, which is not in accordance with the C1G2 standard that requires a tag to pick exactly one slot in a frame. However, this assumption does not create any problems because the expected number of slots that a tag chooses in a frame is still one. The analysis with this assumption is, therefore, asymptotically the same as that without this assumption. Bordenave *et al.* further explained in detail why this independence assumption in analyzing Aloha based protocols provides results just as accurate as if all the analysis was done without this assumption [2]. This independence assumption is made only to make the formal development tractable. In all our simulations, a tag chooses exactly one slot at the start of a frame. Table I lists the symbols used in this paper.

TABLE I. SYMBOLS USED IN THE PAPER

Symbol	Description
$\mathbb{E}$	set of IDs of tags that need to be monitored
$\mathbb{E}_p$	tags in $\mathbb{E}$ that are present in population
$\mathbb{U}$	set of unexpected tags
α	required reliability
m	# of tags in $\mathbb{E}$ missing from population
T	threshold to detect missing tags
f_i	frame size of the i^{th} frame
f	asymptotic value of the frame size
n	asymptotic value of # of frames
R_i	seed used by the controller in i^{th} frame
$h(.)$	hash function used by tags to select slot
p_i	persistence probability used in the i^{th} frame
p	asymptotic value of persistence probability
k_i	# of slots that are 1 in the i^{th} pre-computed frame
q	prob. that at least one present tag selects a slot
g	prob. that a missing tag event is detected
S_d	total # of slots for RUN
Z	rand. var. for # of frames a tag participates in
D	rand. var. for slot of first detectetion
X_{ij}	random var. for the event that the j^{th} 0 slot in the i^{th} pre-computed frame is 1 in the i^{th} executed frame
$\mathcal{N}_i^{01}$	random var. for # of 0 slots in the i^{th} pre-computed frame that are 1 in the i^{th} executed frame

IV. Protocol Description

To detect if any of the tags in set $\mathbb{E}$ is missing from the population, in RUN, the central controller executes up to n Aloha frames using the RFID readers. There are 6 steps involved in executing each frame. First, before executing any frame i, the controller calculates the optimal values of frame size f_i, persistence probability p_i, and generates a random seed number R_i. Second, as the controller knows the IDs in set $\mathbb{E}$, it *pre-computes* which tag in $\mathbb{E}$ will choose which slot in the i^{th} frame. Thus, it knows which slots of the executed i^{th} frame should be nonempty if all the tags in $\mathbb{E}$ were present and a single reader covered the entire population. It represents the nonempty slots in the pre-computed frame with 1s and all other slots with 0s. Third, it provides each reader with the parameters f_i, p_i, and R_i and asks each of them to execute the i^{th} frame using these parameters. The motivation behind using the same values of f_i, p_i, and R_i across all readers for the i^{th} frame is to enable RUN to work with multiple readers with overlapping regions. As all readers use the same values of f_i, p_i, and R_i in the i^{th} frame, the slot number that a particular tag chooses in the i^{th} frame of each reader covering this tag is the same *i.e.*, $h(f_i/p_i, R_i, ID)$ evaluated by the tag results in same value for each reader. Fourth, each reader executes the frame on its turn as per the reader scheduling protocol and sends the responses in the frame back to the controller. Fifth, when the controller receives the i^{th} frame of each reader, it applies logical OR operator on all the received i^{th} frames and obtains a resultant ORed frame. This resultant ORed frame is same as if received by a single reader covering all the tags. Sixth, the controller compares all the slots in the pre-computed i^{th} frame with the corresponding slots in the resultant ORed i^{th} frame. If there is any slot that is 1 in the pre-computed frame but 0 in the resultant ORed frame, the controller detects this as a missing tag event because such a slot implies that all tags in $\mathbb{E}$ that mapped to this slot in the pre-computed frame are absent from the population. At this point, the controller stops the protocol and does not execute the remaining $n-i$ frames. If the controller does not detect a missing tag event even after each reader has executed n frames, it declares that the number of missing tags m is less than the threshold T.

V. Parameter Optimization

Recall from the previous section that before executing any frame i, the controller calculates the optimal values of frame size f_i and persistence probability p_i. For this, the controller first estimates the value of $|\mathbb{U}|$ at the start of the i^{th} frame, represented by $|\tilde{\mathbb{U}}_i|$, based on the responses from the tag population in the previous $i-1$ frames. Details about estimating the value of $|\mathbb{U}|$ will be given in Section V-A. Then, using this estimate along with the values of $|\mathbb{E}|$, α, and T, the controller calculates the optimal values of the frame size f_i and persistence probability p_i such that RUN achieves the required reliability in shortest time. Before asking the readers to execute

the i^{th} frame, the controller also recalculates the maximum number of frames that it should execute, represented by n_i. As the controller executes more and more frames, *i.e.*, as i increases, the estimate $|\tilde{\mathbb{U}}_i|$ asymptotically becomes equal to $|\mathbb{U}|$. Consequently, f_i, p_i, and n_i asymptotically become equal to constants f, p, and n, respectively. When the estimate of $|\mathbb{U}|$ does not change by more than 1% in 10 consecutive frames, the controller considers the estimate to be close enough to $|\mathbb{U}|$. At this point, the controller calculates the values of f_i, p_i, and n_i and puts $f = f_i$, $p = p_i$, and $n = n_i$, and uses these fixed values of f and p to execute subsequent frames until the total number of frames executed since the first frame become equal to n. Note that the controller executes n frames only if it does not detect any missing tag event in any frame. Otherwise, it terminates the protocol as soon as it detects a missing tag event. For the first frame, *i.e.*, when $i = 1$, the controller uses $f_1 = 2 \times |\mathbb{E}|$, $p_1 = 1$, and $n_1 = \infty$. The choices of the values of f_1, p_1, and n_1 are arbitrary and do not really matter because as the controller executes more frames, the frame size, the persistence probability, and the number of frames converge to constants f, p, and n, respectively.

In rest of this section, we will derive equations that the controller uses at the start of each frame to calculate the optimal values of frame size f, number of times the frames should be repeated n, and persistence probability p to minimize the execution time of RUN while ensuring that its actual reliability is no less than the required reliability. We have dropped the subscript i from these parameters to make the presentation simple. To calculate these optimal values, the controller requires the estimate of $|\mathbb{U}|$. Next, we will first present a method to obtain this estimate at the start of any frame i based on the responses from the tag population in the previous $i-1$ frames. Second, using the estimate of $|\mathbb{U}|$, we will derive an expression for the false positive probability, *i.e.*, the probability that a missing tag is detected as present. Third, we will use the expression for false positive probability in conjunction with the required reliability α and threshold T to obtain an equation with three unknowns f, p, and n. To ensure that the actual reliability is greater than or equal to the required reliability, the controller must use the values of f, p, and n that satisfy this equation. We call this equation the *reliability condition*. Fourth, we will derive an expression for the total execution time of RUN and minimize it with respect to n to get an expression involving p and n. The controller simultaneously solves this expression with the reliability condition using $p = 1$ to obtain the optimal values of f and n. Last, we will show how to bring the value of f within limit when the optimal value of the frame size exceeds the C1G2 specified upper limit of 2^{15}, We will also calculate the expected number of slots RUN takes to detect the first missing tag event. Next, we describe these five steps in detail.

A. Estimating Number of Unexpected Tags

In this section, we present a method to estimate the number of unexpected tags in the population at the start of any frame i. Although a lot of work has been done by the research community to estimate the number of tags present in an RFID tag population [3], [5], [6], [12], there is no work on estimating the size of some subset of RFID tag population. In our case, that subset is the set of unexpected tags in the population.

Recall from Section IV that in any frame i, the slots that are 0 in the i^{th} pre-computed frame are the slots that only the tags in set $\mathbb{U}$ can select when the reader executes the i^{th} frame. This is because we have prior knowledge that the tags in set $\mathbb{E}$ will select only those slots in the i^{th} executed frame that are 1 in the i^{th} pre-computed frame. The intuition behind our estimation method is that as the number of unexpected tags in a population increases, the number of slots that are 0 in a pre-computed frame but are 1 in the corresponding executed resultant ORed frame also increase. The number of such slots in any given frame is a function of $|\mathbb{U}|$ and can, therefore, be used to estimate the value of $|\mathbb{U}|$.

Next, we derive an expression that relates the number of slots that are 0 in a pre-computed frame but are 1 in the corresponding executed resultant frame with the value of $|\mathbb{U}|$. We will use this expression to obtain the estimate of $|\mathbb{U}|$. Let the size of the i^{th} frame be f_i and let k_i out of these f_i slots be 1s in the pre-computed frames. Let j be the j^{th} 0 slot in the pre-computed frame. Thus, $1 \leq j \leq f_i - k_i$. Let X_{ij} be an indicator random variable for the event that the j^{th} 0 slot in the i^{th} pre-computed frame turns out to be 1 in the i^{th} executed resultant frame. The expected value of X_{ij} is given by

$$E[X_{ij}] = P\{X_{ij} = 1\} = 1 - \left(1 - \frac{p_i}{f_i}\right)^{|\mathbb{U}|} \approx 1 - e^{-\frac{p_i}{f_i}|\mathbb{U}|}$$

Let $\mathcal{N}_i^{01}$ be a random variable representing the number of slots that are 0 in the i^{th} pre-computed frame but 1 in the i^{th} executed resultant frame. Thus, $\mathcal{N}_i^{01} = \sum_{j=1}^{f_i - k_i} X_{ij}$. As $\{X_{i1}, X_{i2}, \ldots, X_{i(f_i-k_i)}\}$ forms a set of identically distributed random variables, $E[\mathcal{N}_i^{01}]$ is given by

$$E[\mathcal{N}_i^{01}] = E[\sum_{j=1}^{f_i-k_i} X_{ij}] = (f_i - k_i) \times E[X_{ij}]$$
$$= (f_i - k_i) \times (1 - e^{-\frac{p_i}{f_i}|\mathbb{U}|}) \quad (1)$$

Let $\tilde{\mathcal{N}}_i^{01}$ represent the observed value of the number of slots that were 0 in the i^{th} pre-computed frame but 1 in the corresponding executed resultant frame. Replacing $E[\mathcal{N}_i^{01}]$ in the equation above with $\tilde{\mathcal{N}}_i^{01}$ and solving for $|\mathbb{U}|$ gives an estimate of $|\mathbb{U}|$. This estimate is obtained by utilizing the information from the i^{th} frame only. While this estimate may not be accurate, if we use the information from a large number of frames, the estimate will become more accurate. Specifically, we leverage the well known statistical result that the variance in the observed value of a random variable reduces by x times if we take the average of x observations of that random variable. Therefore, to obtain the estimate $|\tilde{\mathbb{U}}_i|$ of $|\mathbb{U}|$ at the start of the i^{th} frame, we obtain an estimate from each of the previous $i-1$ frames and take their average. Solving Equation (1) for $|\mathbb{U}|$ and averaging over past $i-1$ frames, the formal expression for $|\tilde{\mathbb{U}}_i|$ becomes

$$|\tilde{\mathbb{U}}_i| = -\frac{1}{i-1}\sum_{l=1}^{i-1}\frac{f_l}{p_l}\ln\left\{1 - \frac{\tilde{\mathcal{N}}_l^{01}}{f_l - k_l}\right\} \quad (2)$$

Finally, note that the controller obtains this estimate without executing any additional frames. It gets this estimate from the frames it was already executing to detect missing tag events.

B. False Positive Probability

A false positive occurs when all the slots that a particular missing tag maps to in the n pre-computed frames turn out to be nonempty when the frames are executed because some other tags in the population also selected those slots. Lemma 1 gives the expression to calculate the false positive probability.

Lemma 1. *Let m out of $|\mathbb{E}|$ tags be missing, and let there be $|\mathbb{U}|$ unexpected tags in the population. With persistence probability p, frame size f, and number of frames n, the false positive probability, P_{fp}, is given by:*

$$P_{fp} = \left\{1 - p\Big(1 - \frac{p}{f}\Big)^{|\mathbb{U}|+|\mathbb{E}|-m}\right\}^n \tag{3}$$

Proof: The total number of tags in the population are $|\mathbb{U}|+|\mathbb{E}|-m$. Consider an arbitrary tag in $\mathbb{E}$ that is missing from the population. As this tag participates in each pre-computed frame with probability p, it is possible that it does not participate in one or more of the n pre-computed frames. Let Z be the random variable for the number of pre-computed frames in which this missing tag participates. Let q be the probability that a slot that this missing tag maps to in a pre-computed frame is selected by one or more of the tags present in the population in the executed frame. Therefore,

$$P_{fp} = \sum_{z=0}^{n} P\{Z = z\} \times q^z \tag{4}$$

As a missing tag participates in each pre-computed frame with probability p and there are n pre-computed frames, the number of pre-computed frames in which the missing tag participates follows a binomial distribution *i.e.*, $Z \sim \text{Binom}(n, p)$. When a frame is executed, probability that at least one tag in the population chooses the same slot to which the missing tag maps in the pre-computed frame is $1-(1-\frac{p}{f})^{|\mathbb{U}|+|\mathbb{E}|-m}$, which is the value of q. Therefore, Equation (4) becomes

$$P_{fp} = \sum_{z=0}^{n} \binom{n}{z} p^z (1-p)^{n-z} \left\{1 - (1 - \frac{p}{f})^{|\mathbb{U}|+|\mathbb{E}|-m}\right\}^z$$

The binomial theorem states that $\sum_{z=0}^{n} \binom{n}{z} x^z y^{n-z} = (x+y)^n$. Substituting $x = p \times \left\{1 - (1 - \frac{p}{f})^{|\mathbb{U}|+|\mathbb{E}|-m}\right\}$ and $y = 1 - p$, we get Equation (3). ∎

Figure 1 shows the theoretically calculated false positive probability from Equation (3) represented by the solid line and experimentally observed values of false positive probability represented by the dots. To obtain this figure, we use $|\mathbb{E}| = 100$, $|\mathbb{U}| = 500$, $f = 300$, $p = 1$, and $n = 2$. Each dot represents the false positive probability calculated from 100 runs of simulation. We observe that the theoretically calculated values match perfectly with experimentally observed values, showing that our independence assumption that we stated in Section III-D does not cause the theoretical analysis to deviate from practically observed values. We also observe that as the number of missing tags increases, the false positive probability decreases. This means that it is hardest for RUN to detect a missing tag event when $m = T$ and becomes easier as m increases beyond T. Thus, we will use $m = T$ in all further analytical development, because if RUN is able to detect a missing tag event with probability α when $m = T$, it will be able to detect a missing tag event with probability greater than α when $m > T$.

C. Achieving Required Reliability

Following theorem gives the *reliability condition* that the values of f, p, and n need to satisfy in order for RUN to be able to achieve the required reliability.

Theorem 1. *Given a set $\mathbb{E}$ with expected IDs, set $\mathbb{U}$ with unexpected IDs, threshold T, and required reliability α, RUN will achieve the required reliability if the values of f, p, and n satisfy the reliability condition given below.*

$$f = \frac{p(T - |\mathbb{E}| - |\mathbb{U}|)}{\ln\left\{\frac{1-(1-\alpha)^{\frac{1}{nT}}}{p}\right\}} \tag{5}$$

Proof: Probability that RUN detects at least one of the missing tags is $1 - P_{fp}^T$. In the worst case, this probability should at least be equal to α *i.e.*, $1 - P_{fp}^T = \alpha$. Substituting the R.H.S of Equation (3) for P_{fp} gives

$$1-\alpha = \left\{1-p\Big(1-\frac{p}{f}\Big)^{|\mathbb{U}|+|\mathbb{E}|-T}\right\}^{nT} \approx \left\{1-pe^{-\frac{p}{f}(|\mathbb{U}|+|\mathbb{E}|-T)}\right\}^{nT}$$

Rearranging the equation above gives Equation (5). ∎

D. Minimizing Execution Time

Following theorem gives the condition that the values of p and n need to satisfy to make the execution time of RUN minimum under the constraint that it achieves the required reliability.

Theorem 2. *Given a threshold T and required reliability α, the execution time of RUN is minimum under the constraint that it achieves the required reliability if the values of p and n satisfy the following equation:*

$$p = \left\{1 - (1-\alpha)^{\frac{1}{nT}}\right\} \left\{(1-\alpha)^{\frac{(1-\alpha)^{\frac{1}{nT}}}{nT(-1+(1-\alpha)^{\frac{1}{nT}})}}\right\} \tag{6}$$

Proof: Execution time is directly proportional to the total number of slots required to detect the missing tag event because the duration of each slot is the same, typically $300\mu s$ for Philips I-Code RFID reader [10]. Let S_d represent the total number of slots. Thus, $S_d = f \times n$. To ensure that RUN achieves the required reliability, we use the value of f from Equation (5). Thus,

$$S_d = \frac{pn(T - |\mathbb{E}| - |\mathbb{U}|)}{\ln\left\{\frac{1-(1-\alpha)^{\frac{1}{nT}}}{p}\right\}} \tag{7}$$

Figure 2 plots S_d as a function of n. We observe that S_d is a convex function of n. Therefore, optimum value of n exists, represented by n_{op}, that minimizes the total number of slots S_d. To find optimal value of n, we differentiate Equation (7) with respect to n and equate the resulting expression to 0, which gives Equation (6). ∎

At the start of each frame, the controller replaces $|\mathbb{U}|$ with its estimate, puts $p = 1$ in Equation (6), and solves it numerically using Brent's method to obtain the optimal value of number of frames n_{op}. Then it puts $n = n_{\text{op}}$ and $p = 1$ in Equation (5) to get the optimal value of frame size f_{op}. When the controller calculates f_{op} and n_{op} like this at the start of

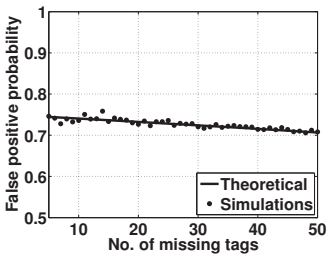


Fig. 1. P_{fp} comparison

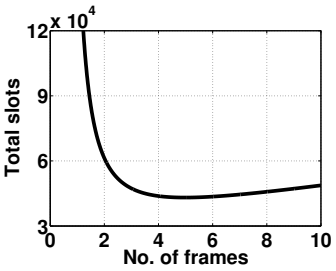


Fig. 2. S_d vs. n

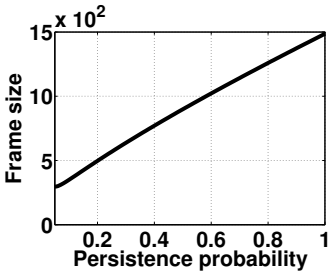


Fig. 3. f vs. p

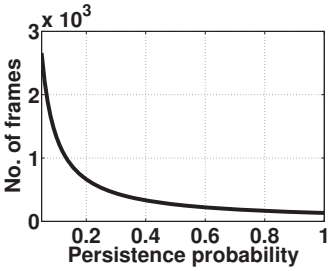


Fig. 4. n vs. p

each frame, the execution time of RUN is minimized. At the same time, as the reliability condition is satisfied, the protocol achieves the required reliability.

E. Handling Large Frame Sizes

For large populations, high required reliability, and/or small threshold, it is possible for the value of f_{op} to exceed the C1G2 specified upper limit of 2^{15}. Next, we describe how we use p to bring the frame size within limits. Bringing the frame size within limits comes at a cost of increased number of slots; greater than the minimum value of S_d that would have been achieved if the controller could use $f_{\text{op}} > 2^{15}$.

When we decrease the value of p, the number of tags that participate in a frame decrease. Therefore, intuitively, the required value of f should also decrease. Figure 3 confirms this intuition. This figure shows the plot of frame size vs. persistence probability, obtained using Equations (5) and (6). We can see that when p decreases, f decreases. Participation by lesser tags means that participation by the tags belonging to both the sets $\mathbb{E}$ and $\mathbb{U}$ decreases. This increases the chances that a given missing tag will not map to any slot in a given pre-computed frame, which means that chances of detecting its absence decrease. Therefore, the overall uncertainty in detection of missing tags increases. To reduce this uncertainty, intuitively, the value of n should increase when p decreases to achieve the required reliability. Figure 4 confirms this intuition. This figure shows the plot of number of frames vs. persistence probability, obtained using Equations (5) and (6). We observe that when p decreases, n increases.

We use these two observations to reduce the value of f whenever $f_{\text{op}} > 2^{15}$. When $f_{\text{op}} > 2^{15}$, the controller uses $f = f_{\text{max}} = 2^{15}$ in Equation (5), which leaves two unknowns, p and n, in the resulting equation. The controller solves the resulting equation simultaneously with Equation (6) to get new values of p and n. The new value of p is less than 1 and the new value of n is greater than n_{op} because $f_{\text{max}} < f_{\text{op}}$. Putting $f = f_{\max}$ in Equation (5) and solving for n, we get

$$n = \frac{\ln\{1-\alpha\}}{T \ln\left\{1 - pe^{\frac{p}{f_{\max}}(T-|\mathbb{E}|-|\mathbb{U}|)}\right\}} \tag{8}$$

Replacing n in Equation (6) with the R.H.S of the equation above, and simplifying, we get

$$\frac{p^2(T-|\mathbb{E}|-|\mathbb{U}|)}{f(e^{\frac{p}{f_{\max}}(|\mathbb{E}|+|\mathbb{U}|-T)} - p)} = \ln\left\{1 - pe^{\frac{p}{f_{\max}}(T-|\mathbb{E}|-|\mathbb{U}|)}\right\}$$

The numerical solution of the equation above gives the new value of p, which the controller puts in Equation (8) to get the new value of n. The controller uses these new values of n and p along with $f = f_{\text{max}}$ to pre-compute the i^{th} frame. Although the total number of slots $S_d = f_{\text{max}} \times n > f_{\text{op}} \times n_{\text{op}}$, this is still the smallest under the constraints that the required reliability is achieved and the frame size does not exceed f_{max}.

F. Expected Detection Time

The values of f and n that we calculate as described in the sections above ensure that in executing n frames, RUN will detect a missing tag event with probability greater than or equal to α if number of missing tags is greater than or equal to T. However, in many cases, the first missing tag event is detected before all n frames are executed. We calculate the expected value of the number of slots that RUN takes to detect the first missing tag event. For this, we calculate the probability that a missing tag event is detected in a given slot and use it to calculate the expected value.

Lemma 2. *Given a set $\mathbb{E}$ with expected IDs, set $\mathbb{U}$ with unexpected IDs, and threshold T, when controller executes RUN with persistence probability p and frame size f, the probability g that a missing tag event is detected in any slot is given by the following equation.*

$$g = \left\{1 - \left(1 - \frac{p}{f}\right)^T\right\} \times \left\{\left(1 - \frac{p}{f}\right)^{|\mathbb{U}|+|\mathbb{E}|-T}\right\} \tag{9}$$

Proof: Probability that a missing tag event is detected in a given slot is the product of the probability that at least one missing tag maps to this slot in the pre-computed frame and the probability that no tag in the population selects that slot in the executed frame. Considering the scenario where it is hardest for RUN to detect a missing tag event *i.e.*, when $m = T$, probability that at least one of the missing tags maps to the given slot in the pre-computed frame is $1 - \left(1 - \frac{p}{f}\right)^T$. The probability that none of the tags present in the population selects that slot is $\left(1 - \frac{p}{f}\right)^{|\mathbb{U}|+|\mathbb{E}|-T}$. The product of these two probabilities gives the expression for g in Equation (9). ■

Following theorem gives the expected value of the number of slots that RUN takes to detect the first missing tag.

Theorem 3. *Let D be the random variable for the slot number when the first missing tag event is detected. Given that the probability of detecting a missing tag event in a slot is g, as calculated in Lemma 2, frame size is f, and number of frames is n, we get*

$$E[D] = \frac{1 - (1-g)^{fn} - fng(1-g)^{fn}}{g} \tag{10}$$

Proof: The random variable D follows geometric distribution with parameter g *i.e.*, $P\{D = i\} = (1-g)^{i-1}g$. The expected value, thus, becomes

$$E[D] = \sum_{i=1}^{S_d} iP\{D = i\} = \sum_{i=1}^{f\times n} ig(1-g)^{i-1}$$
$$= \frac{1-(1-g)^{fn} - fng(1-g)^{fn}}{g}$$

■

VI. Performance Evaluation

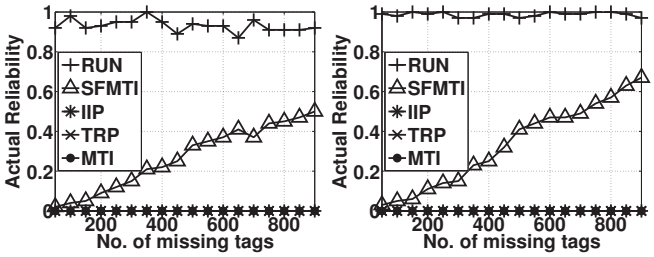
We implemented RUN in Matlab. Although, none of the existing protocols handles the presence of unexpected tags and except for TRP, none of them is C1G2 compliant, we still implemented four prior state of the art missing tag detection protocols in Matlab namely TRP [15], IIP [7], MTI [18], and SFMTI [8], and compared their performance with RUN. We calculated parameter values for these protocols by following the instructions in their respective papers. We also implemented the fastest existing tag collection protocol TH [13]. We choose tag ID length of 64 bits as specified in the C1G2 standard. Note that the distributions of the IDs of expected, unexpected, and missing tags do not matter because RUN is independent of ID distributions.

We first evaluate the actual reliability of RUN and the existing protocols for multiple values of required reliability, keeping the unexpected tag population size fixed and changing the number of missing tags. We also show the time taken by each protocol to detect the first missing tag event. Second, we evaluate the actual reliability of RUN and the existing protocols for multiple values of required reliability by keeping the number of missing tags fixed and changing the unexpected tag population size. We again show the time taken by each protocol to detect the first missing tag event. Third, we study the actual reliability achieved by each protocol when the number of tags missing from the population is different from the value of threshold T. Last, we compare the detection times of our protocols with the fastest tag collection protocol TH.

A. Impact of Number of Missing Tags

RUN is the only protocol that achieves the required reliability in the presence of unexpected tags for any number of missing tags. Figures 5(a) and 5(b) show the actual reliability achieved by RUN and all existing protocols for $\alpha = 0.9$ and 0.99, respectively. These figures are plotted using $|\mathbb{E}| = 1000$, $|\mathbb{U}| = 10000$ and m is varied from 50 to 900. The actual reliabilities are obtained using 100 runs of each protocol for each value of m. None of the existing protocols achieves the required reliability because none of them is designed to handle unexpected tags. Among the existing protocols, SFMTI has the highest actual reliability of up to 0.67

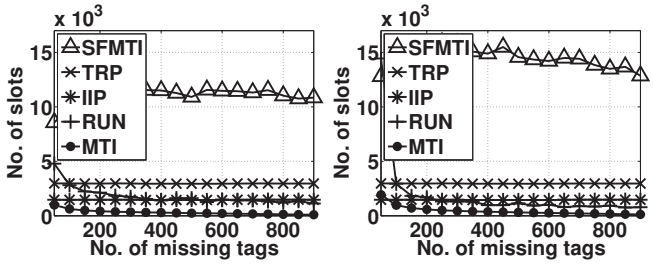
RUN is the fastest protocol that achieves the required reliability compared to the existing protocols. Figures 6(a) and 6(b) show the average times each protocol took to either detect the first missing tag event if it finds a missing tag or to complete execution if it does not find a missing tag. From these figures, MTI seems to have smaller detection time compared to RUN, but when we observe these figures in conjunction with Figures 5(a) and 5(b), we see that the actual reliability of MTI



(a) $\alpha = 0.90$ (b) $\alpha = 0.99$

Fig. 5. Actual reliability vs. number of missing tags

is close to 0, far lower than the required reliability. This shows that for majority of times, MTI completed execution without detecting any missing tags due to the unexpected tags.

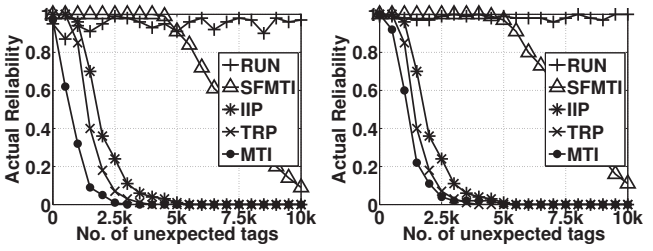


(a) $\alpha = 0.90$ (b) $\alpha = 0.99$

Fig. 6. Detection time vs. number of missing tags

B. Impact of Number of Unexpected Tags

RUN is the only protocol that achieves the required reliability in the presence of unexpected tags while existing protocols achieve the required reliability only when there are no unexpected tags in the population. Figures 7(a) and 7(b) show the actual reliability obtained by RUN and the existing protocols for $\alpha = 0.9$, and 0.99, respectively. These figures are plotted using $|\mathbb{E}| = 1000$, $m = 200$, and $|\mathbb{U}|$ is varied from 0 to 10000. RUN always achieves the required reliability whereas the existing protocols achieve the required reliability only when $|\mathbb{U}|$ is close to zero.

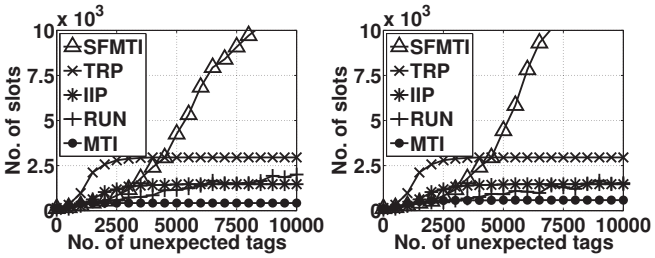


(a) $\alpha = 0.90$ (b) $\alpha = 0.99$

Fig. 7. Actual reliability vs. number of unexpected tags

RUN is the fastest protocol that achieves the required reliability compared to the existing protocols even when there are no unexpected tags in the population. Figures 8(a) and 8(b) show the average times each protocol took to either detect the first missing tag event or complete execution without detecting any missing tags. From these figures, MTI again seems to have smaller detection time compared to RUN when number of unexpected tags in the population is large, but when we analyze these figures in conjunction with Figures 7(a) and 7(b), we see that actual reliability of MTI is close to 0 when number of unexpected tags in the population is large. Figures 7(a) and

7(b) show that SFMTI achieves the required reliability for up to 5000 unexpected tags, but then Figures 8(a) and 8(b) show that its execution time is 5 times greater than RUN.



(a) $\alpha = 0.90$ (b) $\alpha = 0.99$

Fig. 8. Detection time vs. number of unexpected tags

C. Impact of Deviation from Threshold

The actual reliability of RUN exceeds the required reliability when the number of missing tags in the population exceed the threshold T. This is seen in Figure 9, which plots the actual reliabilities of all protocols when number of missing tags are larger or smaller compared to T. This figure is made using $|\mathbb{E}| = 1000$, $|\mathbb{U}| = 10000$, $T = 200$, $\alpha = 0.99$, and m is varied from 50 to 900. The actual reliability of RUN is less than the required reliability only when the number of missing tags are less than T, but this is insignificant because we are interested in detecting the missing tags only if the number of missing tags in a population exceed the threshold T.

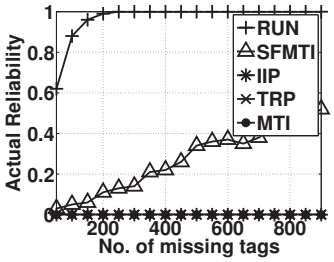


Fig. 9. Effect of difference between m and T

D. Comparison with Tag ID Collection Protocol

RUN is faster than the fastest tag ID collection protocol, TH, in all practical scenarios. For example, for $|\mathbb{E}| = 1000$, $|\mathbb{U}| = 10000$, and $T = m = 200$, RUN is 14.3 times faster than TH for $\alpha = 0.99$. As the threshold T decreases and/or the required reliability increases, detection time of RUN increases. Therefore, there exists a value of T and/or α for a given $|\mathbb{E}|$ and $|\mathbb{U}|$ for which the tag ID collection protocol is faster than RUN. For example, for $|\mathbb{E}| = 1000$, $|\mathbb{U}| = 10000$, and $T = 200$, TH is faster than RUN when required α is greater than 0.99999. Such high values of α are seldom required. Similarly, for $|\mathbb{E}| = 1000$, $|\mathbb{U}| = 10000$, and $\alpha = 0.99$, TH is faster than RUN for $T < 0.001$. In all practical scenarios, the threshold can not be less than 1. Therefore, practically, RUN is always faster than the tag ID collection protocols.

VII. Conclusions

The key technical contribution of this paper is in proposing a protocol to detect missing tag events in the presence of unexpected tags. This paper represents the first effort on addressing this important and practical problem. The key technical depth of this paper is in the mathematical development of the theory that RUN is based upon. The solid theoretical underpinning ensures that the actual reliability of RUN is greater than or equal to the required reliability. We have proposed a technique that our protocol uses to handle large frame sizes to ensure compliance with the C1G2 standard. We have also proposed a method to implicitly estimate the size of the unexpected tag population without requiring an explicit estimation phase. We implemented RUN and conducted side-by-side comparisons with four major missing tag detection protocols even though the existing protocols do not handle the presence of unexpected tags. Our protocols significantly outperform all prior protocols in terms of actual reliability as well as detection time.

Acknowledgement

This work is partially supported by the National Natural Science Foundation of China under Grant Numbers 61472184, 61321491, 61272546, the Jiangsu Future Internet Program under Grant Number BY2013095-4-08, and the Jiangsu High-level Innovation and Entrepreneurship (Shuangchuang) Program.

References

[1] Preliminary national retail security survey findings. https://nrf.com/news/national-retail-security-survey-retail-shrinkage-totaled-345-billion-2011.

[2] C. Bordenave, D. McDonald, and A. Proutiere. Performance of random medium access control, an asymptotic approach. In *Proc. ACM SIGMETRICS*, 2008.

[3] B. Chen, Z. Zhou, and H. Yu. Understanding rfid counting protocols. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 291–302. ACM, 2013.

[4] EPCGlobal Inc. *Radio-Frequency Identity Protocols C1G2 UHF RFID Protocol for Communications at 860 MHz–960 MHz*, 2013.

[5] M. Kodialam and T. Nandagopal. Fast and reliable estimation schemes in RFID systems. In *Proc. MobiCom*, pages 322–333, 2006.

[6] M. Kodialam, T. Nandagopal, and W. C. Lau. Anonymous tracking using RFID tags. In *Proc. IEEE INFOCOM*, 2007.

[7] T. Li, S. Chen, and Y. Ling. Identifying the missing tags in a large RFID system. In *Proc. MobiHoc*, pages 1–10, 2010.

[8] X. Liu, K. Li, G. Min, Y. Shen, A. Liu, and W. Qu. Completely pinpointing the missing RFID tags in a time-efficient way. *IEEE Transactions on Computers*, pages 1–11, 2013.

[9] W. Luo, S. Chen, T. Li, and Y. Qiao. Probabilistic missing-tag detection and energy-time tradeoff in large-scale RFID systems. In *Proc. MobiHoc*, pages 95–104, 2012.

[10] Philips-Semiconductors. SL2 ICS11 I.Code UID smart label IC functional specification datasheet.

[11] M. Roberti. A 5-cent breakthrough. *RFID Journal*, 5(6), 2006.

[12] M. Shahzad and A. X. Liu. Every bit counts - fast and scalable RFID estimation. In *Proc. ACM MobiCom*, pages 365–376, 2012.

[13] M. Shahzad and A. X. Liu. Probabilistic optimal tree hopping for RFID identification. In *Proc. ACM SIGMETRICS*, 2013.

[14] A. D. Smith, A. A. Smith, and D. L. Baker. Inventory management shrinkage and employee anti-theft approaches. *International Journal of Electronic Finance*, 5(3):209–234, 2011.

[15] C. C. Tan, B. Sheng, and Q. Li. How to monitor for missing RFID tags. In *Proc. IEEE ICDCS*, pages 295–302, 2008.

[16] S. Tang, J. Yuan, X.-Y. Li, G. Chen, Y. Liu, and J. Zhao. Raspberry: A stable reader activation scheduling protocol in multi-reader RFID systems. In *Proc. IEEE ICNP*, pages 304–313, 2009.

[17] H. Vogt. Efficient object identification with passive RFID tags. *Pervasive Computing*, 2002.

[18] R. Zhang, Y. Liu, Y. Zhang, and J. Sun. Fast identification of the missing tags in a large RFID system. In *Proc. IEEE SECON*, 2011.

[19] B. Zhen, M. Kobayashi, and M. Shimizu. Framed ALOHA for multiple RFID objects identification. *IEICE Trans. on Communications*, 2005.